Robot Programming #8

아날로그 데이터 출력

Dept. of Mech. Robotics and Energy Eng.
Dongguk University

Concepts of DA conversion

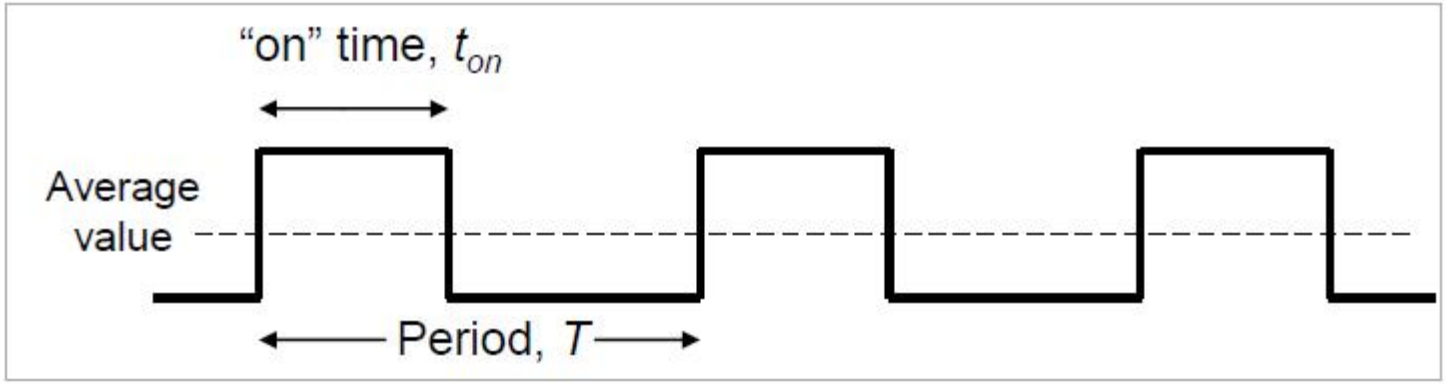
- We can represent the digital-to-analog convertor (DAC) as a block diagram with a digital input, D, and an analog output, vo.
- The Arduino does not have a built-in digital-to-analog converter (DAC), but it can pulse-width modulate (PWM) a digital signal to achieve some of the functions of an analog output.
- The function used to output a PWM signal is analogWrite(pin, value). pin is the pin number used for the PWM output. value is a number proportional to the duty cycle of the signal.

Pulse Width Modulation(PWM)

- What is Pulse Width Modulation?
- Pulse width modulation (PWM) is a simple method of using a rectangular digital waveform to control an analog variable.
- PWM control is used in a variety of applications, ranging from communications to automatic control
- The period is normally kept constant, and the pulse width, or “on” time is varied

Pulse Width Modulation(PWM)

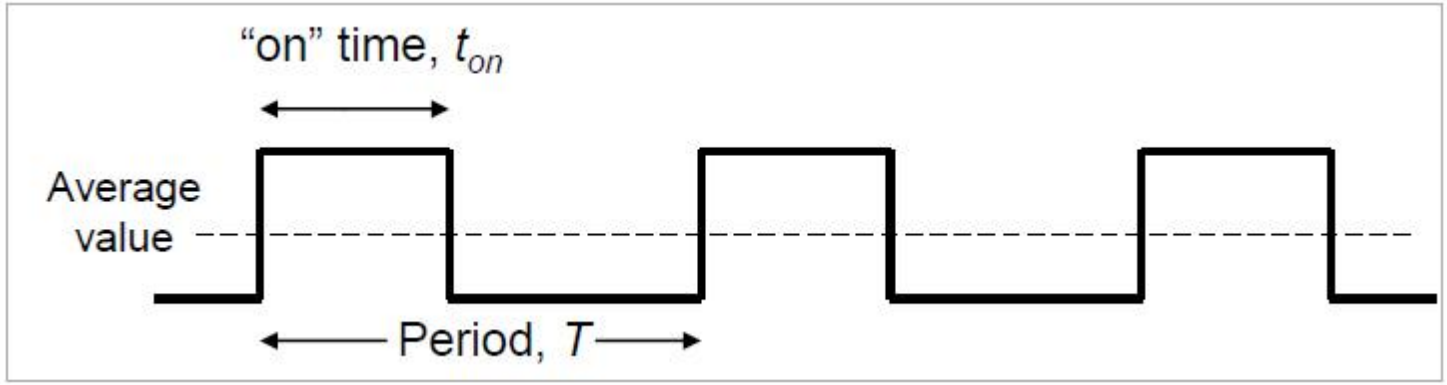
- Pulse width modulation (PWM) is a simple method of using a rectangular digital waveform to control an analog variable.
- PWM signal is controlled by the duty cycle, that is the proportion of time that the pulse is ‘on’ or ‘high’, and is expressed as a percentage:



Pulse Width Modulation(PWM)

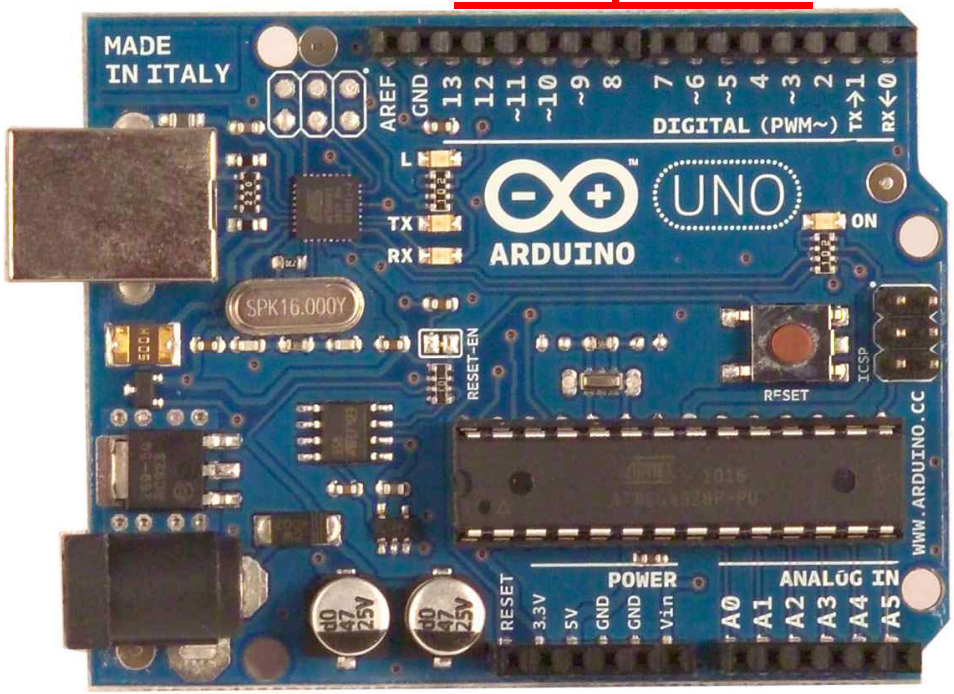
- By controlling the duty cycle, we control this average value.
- If the on time is small, the average value is low; if the on time is large, the average value is high.

duty cycle = 100% * (pulse on time) / (pulse period)



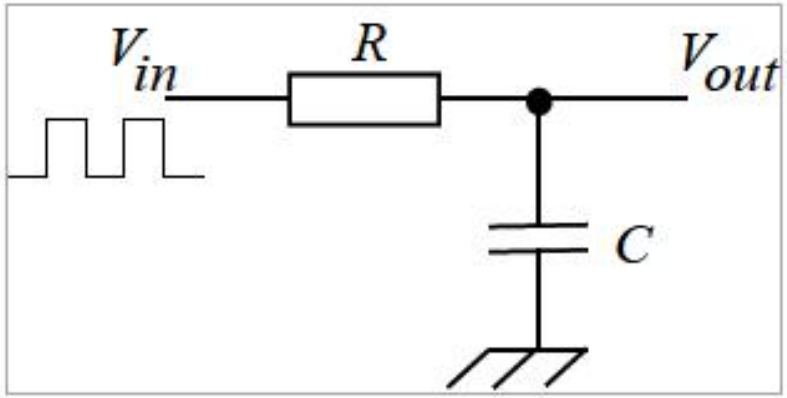
Arduino Uno

Pins(3,5,6,9,10,11) with ~ are PWM
[Analog Output]



Pulse Width Modulation(PWM)

- The average value can be extracted from the PWM stream with a low-pass filter
- In this case, and as long as PWM frequency and values of R and C are appropriately chosen, Vout becomes an analog output
- In practice, this sort of filtering is not always required; many physical systems have response characteristics which, in reality, act like low pass filters



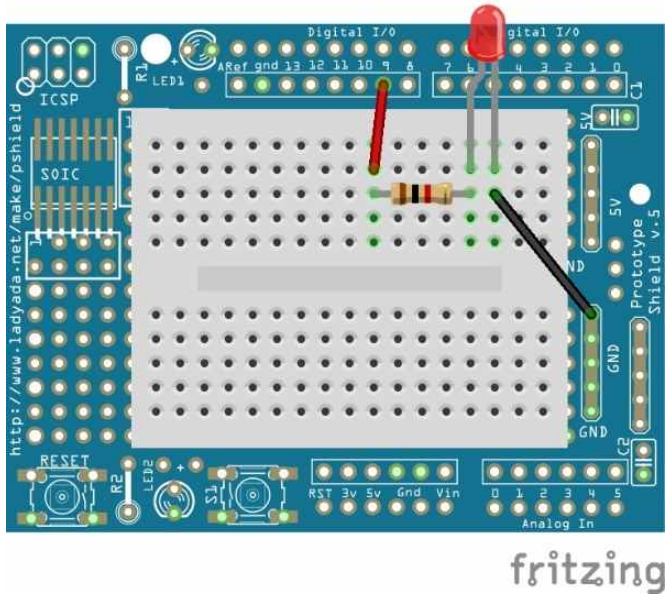
Applications using PWM

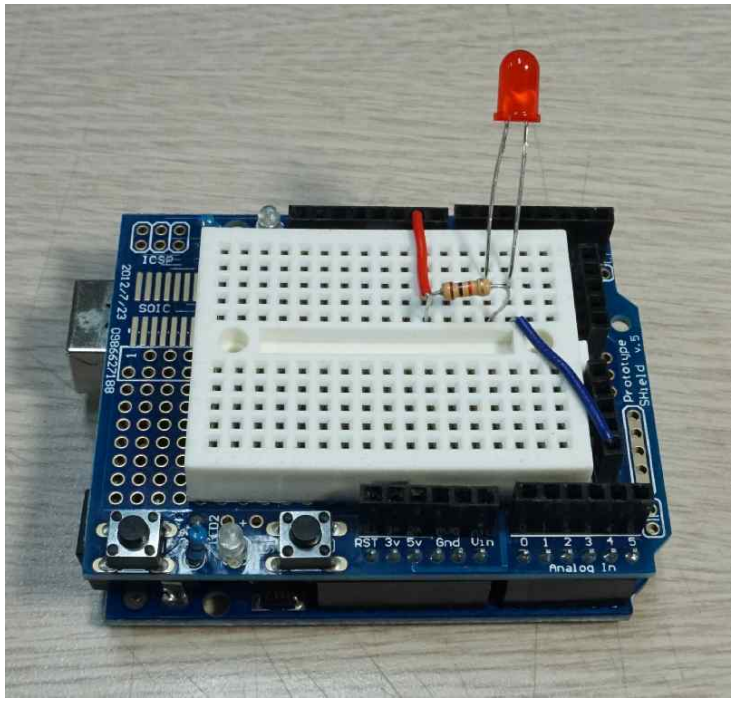
- Devices used in robotics
  - DC motors
  - Servos
  - Solenoids
  - Closed loop control systems
  - Communications and pulse code modulation

- Benefits include
  - Microprocessor control
  - Efficient use of power
  - Tolerance to analog noise
  - Not susceptible to component drift

PWM Output

- Let’s connect the LED to pin 9 of the Arduino.





PWM Output

- Program:

```
void setup() {
 pinMode(9, OUTPUT);
}

void loop() {
  for(float y=0; y<=1.0; y=y+0.25)
  {
    analogWrite(9, y*255);
    delay(500);
  }

}
```

A Closer Look

- The brightness of the LED is controlled by analogWrite().

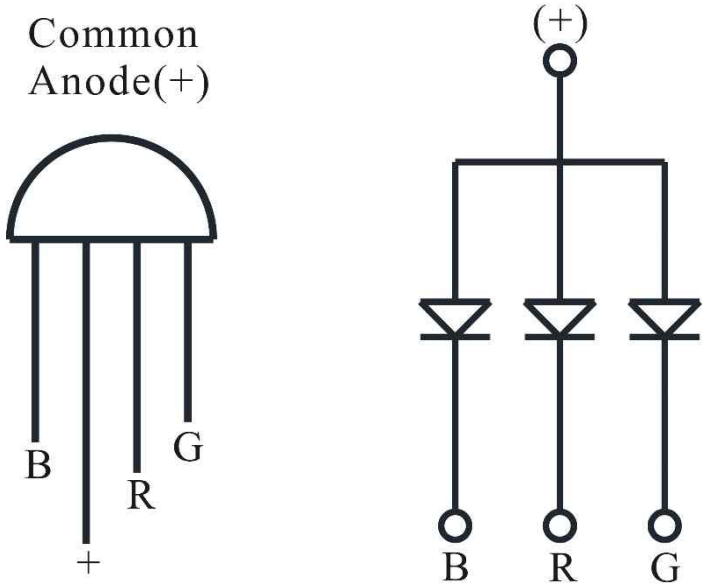
```
analogWrite(9,i);
```

RGB LED

- An RGB LED isn't a single LED.
- It's actually three LEDs in one small package: one Red, one Green and one Blue.
- When you turn them on their light mixes together and you get other colors.
- The color you get is a result of the intensity of the individual red, green and blue LEDs.
- We control the intensity with PWM which we've used before to control LED brightness.

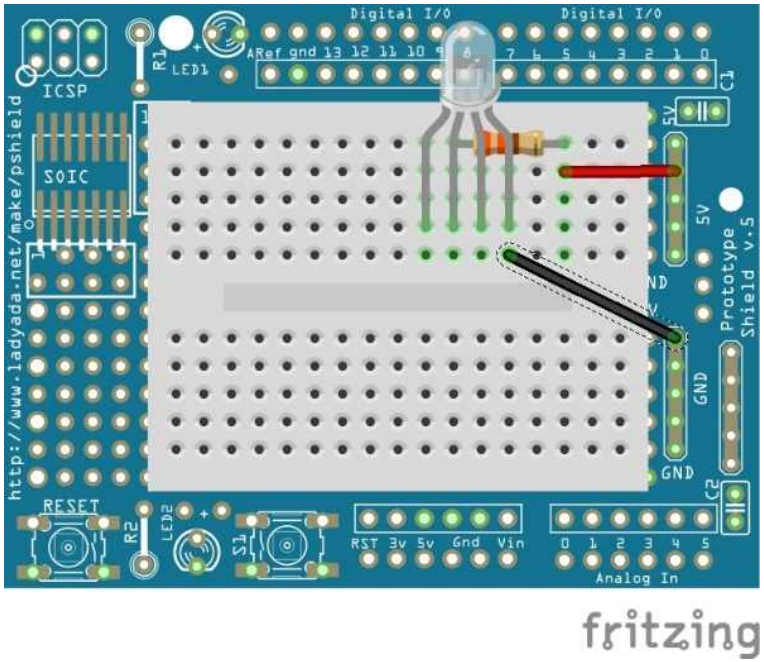
RGB LED

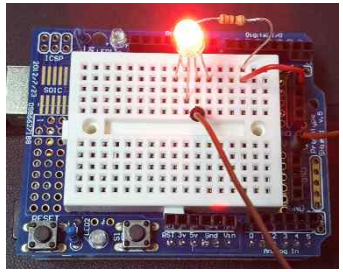
- There are two types of the RGB LED: Cathode and Anode types.
- We will use Anode type in our experiment.

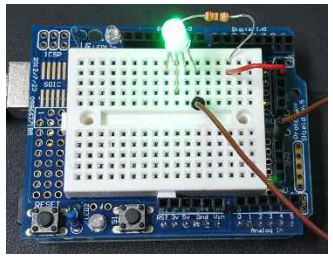


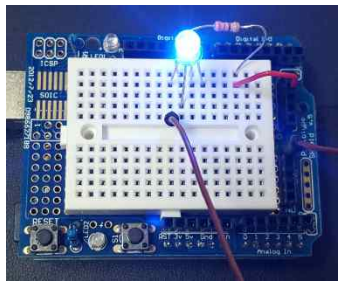
RGB LED

- Let's test the RGB LED before connecting to the Arduino pin. RGB pins need to be connected to ground(LOW).



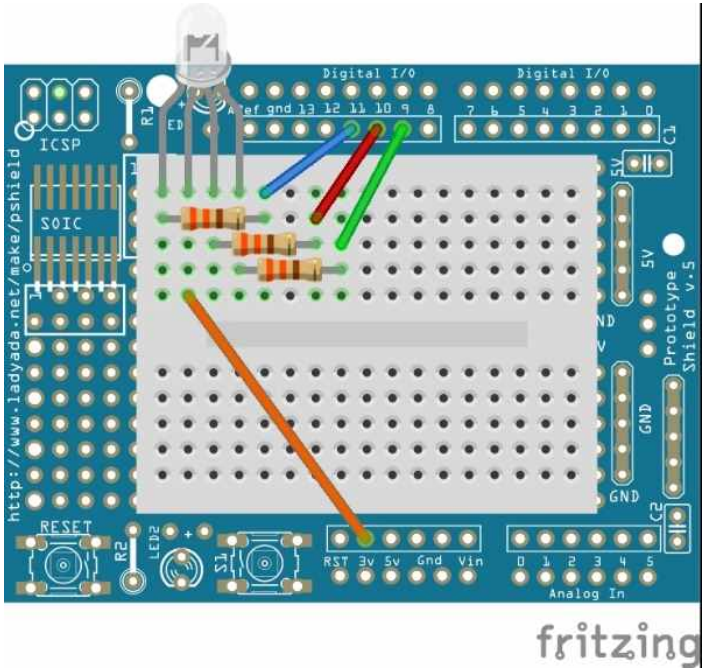


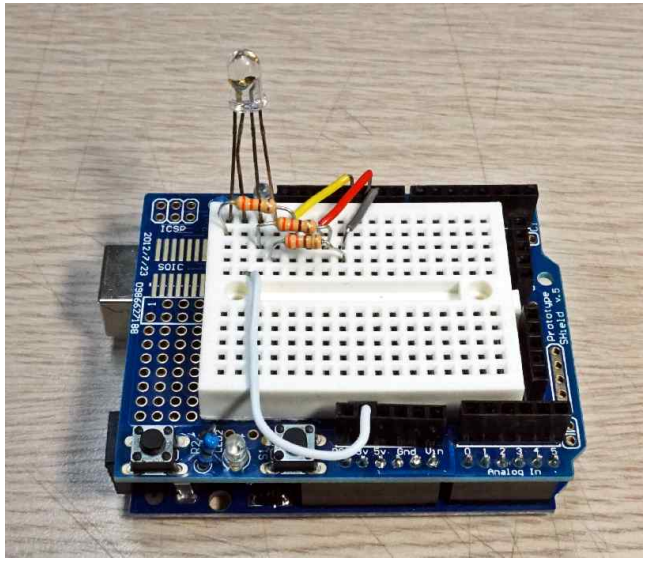




RGB LED

- Connect common(long leg) to 3V of the Arduino, Blue to p11, Red to p10 and Green to p9 as shown below.





RGB LED

- Write the following code and test the color.

```
int green = 9;
int red = 10;
int blue = 11;

void setup() {
  pinMode(green, OUTPUT);
  pinMode(red, OUTPUT);
  pinMode(blue, OUTPUT);
}

void loop() {
  digitalWrite(green, 0); digitalWrite(red, 1);
  digitalWrite(blue, 1);
  delay(500);
  digitalWrite(green, 1); digitalWrite(red, 0);
  digitalWrite(blue, 1);
  delay(500);
  digitalWrite(green, 1); digitalWrite(red, 1);
  digitalWrite(blue, 0);
  delay(500);
}
```